High Performance Computing Software

JPL Internal Seminar Series

Common Component Architecture (CCA) Experiences and Measurements

Dr. Daniel S. Katz

Thursday, July 3, 2003
12:00 noon – 1:00 p.m.
Building 126, Room 225

A growing problem in the development of large-scale multi-disciplinary scientific applications for high-performance computers is managing the interaction between portions of the application developed by different groups, possibly at different periods if code-reuse is desired. In the business world, component-based software engineering has been proposed as a solution. These technologies may not be appropriate for scientific computing. To examine this issue, the Common Component Architecture (CCA) Forum was formed. It is developing a component architecture specification to address the unique challenges of high-performance scientific computing, with emphasis on scalable parallel computations that use possibly distributed resources, and developing a reference framework, various components, and supplementary infrastructure.

NASA's ESTO-CT (Earth Science Technology Office's Computation Technologies) project has so far been successful in achieving its goal of "Demonstrating the power of high-end, scalable, and cost-effective computing environments to further our understanding and ability to predict the dynamic interaction of physical, chemical, and biological processes affecting the Earth, the solar-terrestrial environment, and the universe". However, the impact on software development for most scientists in the broader community has been limited. This was recognized by NASA management, and the project's current work emphasizes frameworks and interoperability for large-scale high performance scientific software development and maintenance. This seminar is a result of the ongoing study of the CCA Forum's technology by the ESTO-CT project.

We will discuss qualitative and quantitative examinations of the CCA software as applied to sequential and parallel example applications that include unstructured adaptive mesh refinement (AMR). The reason for this choice is that AMR libraries are used by many of the ESTO CT applications, which are generally physical, grid- or mesh-based simulations. The process of modifying existing Fortran 90 code consisting of a driver routine and an AMR library into two components will be described in detail, and the performance of the original application, and the componentized version will be compared. We will also discuss parallel components, including the procedure used to transform the code into components, as well as comparing the performance of the two versions.

For questions, please contact Hans Zima at 4-8980.